Encrypted HTTP/2 Traffic Monitoring: Standing the Test of Time and Space

Pierre-Olivier Brissaud, Jérôme François, Isabelle Chrisment, Thibault Cholez, Olivier Bettan

▶ **To cite this version:**

Pierre-Olivier Brissaud, Jérôme François, Isabelle Chrisment, Thibault Cholez, Olivier Bettan. Encrypted HTTP/2 Traffic Monitoring: Standing the Test of Time and Space. WIFS2020 - IEEE International Workshop on Information Forensics and Security, Dec 2020, New-York/Virtual, United States. 10.1109/WIFS49906.2020.9360895 . hal-03032578

HAL Id: hal-03032578
https://hal.science/hal-03032578

Submitted on 30 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Encrypted HTTP/2 Traffic Monitoring: Standing the Test of Time and Space

Pierre-Olivier Brissaud *†, Jérôme François *, Isabelle Chrisment *, Thibault Cholez *, Olivier Bettan †,
* Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
Email: {jerome.francois, isabelle.chrisment, thibault.cholez}@loria.fr
† Thales, Palaiseau, France
Email: {pierre-olivier.brissaud, olivier.bettan}@thalesgroup.com

***Abstract*—Encrypted HTTP/2 (h2) has been worldwide adopted since its official release in 2015. The major services over Internet use it to protect the user privacy against traffic interception. However, under the guise of privacy, one can hide the abnormal or even illegal use of a service. It has been demonstrated that machine learning algorithms combined with a proper set of features are still able to identify the incriminated traffic even when it is encrypted with h2. However, it can also be used to track normal service use and so endanger privacy of Internet users. Independently of the final objective, it is extremely important for a security practitioner to understand the efficiency of such a technique and its limit. No existing research has been achieved to assess how generic is it to be directly applicable to any service or website and how long an acceptable accuracy can be maintained.**

This paper addresses these challenges by defining an experimental methodology applied on more than 3000 different websites and also over four months continuously. The results highlight that an off-the-shelf machine-learning method to classify h2 traffic is applicable to many websites but a weekly training may be needed to keep the model accurate.

***Index Terms*—Traffic Analysis, HTTP2, HTTPS monitoring, Privacy, TLS, Encrypted Traffic**

I. Introduction

HTTPS is nowadays the default protocol for 59.4%[1] of websites in the Internet mostly due to the users' expectation regarding privacy and security after the scandals revealed over the last decade and the increase of data leakage and identity theft impacting individuals.

Among the websites served by HTTPS, according to W3Techs[2], 41.4% run over HTTP/2 and this number has increased by more than 30% over one year. Moreover, 77 websites of the top 100 websites[3] use HTTP/2, which makes HTTP/2 the future protocol for encrypted web traffic and that will replace HTTP/1.1 over TLS. In this paper, we thus consider the analysis of HTTP2 traffic over TLS also called h2.

Thanks to large deployment of HTTPS and under the guise of privacy, malicious users can also unfortunately hide abnormal or even illegal use of a service. In our prior work [1], we have demonstrated that machine learning algorithms can help fight against this kind of malicious behavior. We have proposed a classification approach which, combined with a proper set of features, is able to identify the incriminated HTTP/2 traffic even when encrypted. It can be used to detect specific forbidden user actions such as a keyword search performed on some services. Quite rightly, we cannot exclude that our technique can be exploited by malevolent users to track user activities and so it indirectly increase the privacy risks. In all cases, the scope of the applicability of the technique and its limit are important to understand. On one hand, a security practitioner may rely on it to trigger alerts and need to know how accurate the technique is in different scenarios. On the other hand, a user expecting privacy would need to know how much is compromised, that may possibly lead to new protocol refinements.

The robustness of a classification model is always questionable and is mainly impacted in our case by two main factors. First, the traffic generated for a same request can change over time (how and what data are retrieved) and thus impacting the validity of models learned. Second, the generalization of a model can only be validated thanks to a large number of different instances to be tested. We refer to these two problems as test-of-time and test-of-space respectively. A practical limitation to address these issues is the availability of extensive datasets (in time and space).

This paper thus extends our previous work [1] by investigating the two questions mentioned above. Besides, as we consider the *de facto* standard for web nowadays, our reported results aim at evaluating the viability of our technique on-the-shelf, i.e. with no specific parameter customization, and at large scale. Therefore it assesses the viability of monitoring encrypted HTTP2.0 traffic.

To summarize our main contributions are three-fold:

- Define a test-of-time and test-of-space methodology for HTTPS classification including the specification of a crawling campaign to collect relevant datasets. The collected dataset can be accessible on-demand (its size is too large to be publicly accessible on our servers). They consists of the pcap and HTML files but also screenshots of each accessed website.
- Evaluate our H2 classifier over around four months (test-of-time) and the impact of a regular re-training.

[1]https://w3techs.com/technologies/details/ce-httpsdefault
[2]https://w3techs.com/technologies/details/ce-http2/all/all
[3]results from our own tests

- Evaluate our H2 classifier on more than 3000 websites (test-of-space).

The paper is organized as follows. Section II presents related works and background, in particular our classification technique omitting in-depth details available in [1]. Our test methodology is given in Section III along with dataset descriptions. Section IV and Section V respectively give the evaluation over time and over different services. Section VI concludes the paper.

II. Related Work and Background

A. Related Work

Privacy is not guaranteed by encryption especially with HTTPS. Already in 1996, Wagner *et al.* [2] found vulnerabilities for SSL protocol against traffic analysis. The encrypted traffic analysis topic is a wide research area and has been approached from different angles.

Velan *et al.* [3] present in their survey classification methods which allow detecting applications or protocols inside a encrypted flow. On their side, Shbair *et al.* [4] proposed a framework for detecting web services over HTTPS.

Pescape *et al.* [5] showed they could distinguish the anonymized networks (Tor, i2P, JonDonym), from each other and to classify types of applications. Website fingerprinting on anonymous networks such as Tor has been extensively studied in literature like in [6]–[9]. These authors used different clustering methods and features, mostly based on the packet size distribution, for their fingerprints.

Another part of the research activities has also concerned mobile traffic analysis for identifying the services installed on a mobile device. AppScanner [10] profiles 110 mobile applications with an accuracy between 73% and 96%. In this area, particular activities or behaviors might be detected within the flow [11]–[13]. On top of that, specific analysis for some applications have been put forward, e.g. for Netflix [14] or Skype [15], [16]. These works are built on the awareness of the encoding techniques for video and audio and their impact on the encrypted domain.

Finally we designed an HTTP/2 traffic classifier (H2CLassifier) [1] in order to detect specific behaviors inside a predefined service over HTTP/2. To the best of our knowledge, H2Classifier is the first solution able to identify user actions on a service protected by HTTPS (with HTTP/2). In this paper we evaluate the H2Classifier over time and space.

For the case of the space it is hard to compare with the other works because we don't detect the same concept. But for the time evaluation it can be tested for different works. However, this question is not usually discuss in the paper presented before. In the work of Juarez [17] which discuss about some limitations of the traffic analysis the topic of efficiency over the time is handle. Especially they show that in their case the accuracy drop by 50% in under one month for their classification. For this reason we deal with this question in this paper to evaluate the impact and find solutions.

B. Traffic Classification: H2Classifier

H2Classifier's goal is to passively and transparently infer actions of encrypted web services, more especially HTTPS using HTTP2. In this work, user actions are actually requests performed by users such as making use of search fields.

Our method leverages the Random Forest (RF) algorithm [18]. For each individual service s in the set S of all services to be monitored, our method will automatically learn a classification model related to particular actions defined a priori. As an example S could be defined as $\{Google\ Images, Amazon, Instagram\}$.

In a nutshell, our technique relies on collected traffic associated with particular actions on a given service s. This traffic contains multiple instances of the same action in order to apply supervised learning with Random Forest and then be able to classify a new unknown instance of collected traffic. Therefore, it is necessary to establish the list of monitored user actions in order to generate and collect the associated traffic, which also prevent us to detect any other (private) actions but the monitored ones.

The main issue of all machine learning-based techniques is the definition of features specific to the addressed problem. In our case, we have to identify the user request and so the content of a request that is contained in TLS records. Due to encryption, only meta-data features can be extracted and helpful [1]:

- Connection statistics (3 features): Number of incoming and outgoing TLS records, total number of bytes exchanged at the TLS layer.
- Burst statistics (10 features): the minimum, maximum, standard deviation, mean and median of (1) number of bytes sent by the server between two client packets and (2) number of records send by the server between 20 records.
- Number of distinct sizes of incoming and outgoing TLS records (two features).
- Top 20 most representative sizes of TLS records (2×20): the 20 TLS record sizes with the lowest frequencies, from incoming and outgoing packets.
- Size distribution: all TLS record size frequencies for both the request and response. It is a theoretically highly dimensional vector (up to $2\times18{,}432$ by definition) but in practice, most values are never observed and have no impact on the result and are so discarded from the feature list.

The rational behind this feature set is to capture indirectly some estimates about the size of loaded objects after a web page is requested, as for example images. These object sizes are actually very discriminating due to their high variance in size and so helpful to fingerprint the content of a page, especially if it is composed of many objects. As an example, thumbnails corresponding to products in an online store are usually related to the user search among other criteria such as her own history and profile. Finally, the feature set is

independent of TLS version (1.2 or 1.3) making *H2Classifier* compatible with last standards.

III. Methodology and Datasets

A. Methodology

In our previous paper [1], the results demonstrate a high accuracy of our technique ($> 94\%$) for five major services in the Internet namely Google Search, Google Images, Google Maps, Amazon and Instagram. However, the robustness of our technique might be questionable, especially to verify how much accurate our technique applies to other web services (test-of-space) and how long the learned RF models can be valid (test-of-time). Indeed, contents in the Internet can be frequently updated; a database of meaningful models (for each service to be monitored) must thus be maintained. This would require to collect traffic representative of the new contents and retraining the models. As an additional question, knowing the root causes of the classifier errors is important in order to assess if it is due to a content change or a modification of the technical environment (protocol version, server software, application libraries used, etc.).

To address these objectives without a bias, the configuration of parameters strictly follows recommendations made in our previous work. For RF classification hyper-parameters, the number of trees and the maximum depth of trees are set to 400 and 50 respectively. Only the 300 most representative features are used based on the Gini importance (identify the features which mostly reduce the impurity based on the Gini criterion).

For evaluating the robustness of the classifier over time and space, two datasets have been constituted and are detailed in Sections III-C and III-D. We make them available on demand[4] to allow reproducible research and benefit to other works. Each dataset includes all the packets of every captured web page as well as the related screenshot, the HTML export and some additional information.

B. Crawling tool

Our data crawler is implemented in Python based on Selenium[5] coupled with geckodriver[6] in order to control a Firefox web browser. This setup allows the automated loading of HTTPS pages while the tcpdump[7] program is used in parallel to capture all the packets. In the following of the paper we refer to packets captured from the loading of a page on a service as a trace.

Our tool is able to automaticaly detect and fill-out a search bar and thus to automate the search of keywords on random websites. During the first loading of a web page, it extracts the HTML of the page and looks for an input tag of type "text", "search" or "url". Most of the time, a search bar has a special attribute name='q'[8][9] but we also consider the attribute names *search*, *research* or *searchbar*.

TABLE I: Temporal dataset

Test-of-time	
Services	Amazon, Instagram, Google, Google Images
Number of keywords per service	500
# traces per keyword per day	4
Number of hosts for capture	1
Total size of dataset	1.8 TB

C. Temporal Dataset

For the temporal evaluation, we actively crawled 500 keywords of four major services (Amazon, Instagram, Google and Google Images). Four individual traces are collected for each couple of keyword and service per day as summarized in Table I. In total, 8000 daily traces are stored during a 121-days period (about 4 months) except a few of maintenance days operated in our lab infrastructure (June 21th to 26th, July 21th and 22th).

For Amazon, Google and Google Images, the list of keywords is identical and was built from the dataset of J. McAuley [19]. It contains a list of exact Amazon product names. From this list we extract the most frequently 3-grams of words and use them as keywords. The benefit of using those keywords is that we get a lot of related results. Regarding Instagram, the top 500 account names sorted by followers have been used. The complete list of the exact keywords used for the different services is not detailed here but is available on the web page of our dataset.

D. Service-wide Dataset

1) Service selection: The service wide-dataset is built by requesting many popular websites. Starting from an initial list of the most visited US websites[10], those with no h2 (HTTPS with HTTP/2) or search bars have been excluded. In addition, a few websites cannot be accessed or generated errors (*e.g.* timeout) in the capture. As a result, our dataset is composed of 3096 websites crawled between July 15th and August 14th 2019. The crawling campaign has been spread over time only for scalability reason by opposition to the previous dataset where time is carefully taken into consideration. Similarly, parallelization was performed thanks to multiple IP addresses. For ethical research reason and avoid flooding the targeted websites, queries have been alternated within a window of 20 websites. Hence, a trace was collected every 15 seconds and a website was queried every 5 minutes in average.

2) Keyword selection: For each visited website, we use 20 keywords and capture 20 traces for each of them (so a total of 400 traces per website) as stated in Table II. Keywords must be carefully selected due to the variety in website content. Irrelevant keywords will lead to meaningless webpages, as for example with no search results whereas our goal is to

[4]http://betternet.lhs.inria.fr/datasets/
[5]https://www.seleniumhq.org/
[6]https://github.com/mozilla/geckodriver
[7]https://www.tcpdump.org/
[8]https://www.w3schools.com/tags/att_input_type_search.asp
[9]https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/search\#Basic_example
[10]https://www.quantcast.com/top-sites/US

TABLE II: Services dataset

Test-of-space	
Number of Services	3096 different
Number of keywords per service	20
Number of traces per keyword	20
Number of hosts for capture	6
Total size of dataset	6 TB

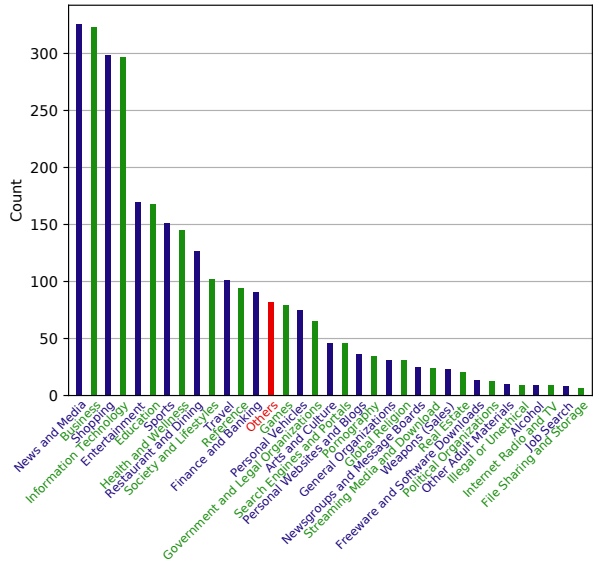

Fig. 1: Categories of the 3096 websites composing our URL list

identify pages associated with particular user actions and so representative of realistic requests.

Thus, we defined a distinct keyword list for each type or category of websites based on the FortiGuard classification [11].

As highlighted in Figure 1, the distribution of the categories of the 3096 tested websites is not balanced. We focused so on defining specific keywords by hand for the 12 most represented categories whereas random words from Oxford English dictionary have been considered in other cases.

IV. Test-of-time

The goal of this experimentation is to evaluate the efficiency of H2Classifier over time and if its accuracy can be maintained by regularly updating the model based on the dataset described in Section III-C.

A. Fixed learned models

The traces collected during the 15 first days are reserved for the training stage of the RF models (one for each service).

Results are presented in Figure 2. The grey areas correspond to maintenance days where no data has been collected. As a recall, 500 keywords are tested four times a day for each service. The value thus represents the mean accuracy, i.e. the proportion of traces that has been correctly classified. The global trend is a slow decrease of the accuracy over the time. This observation acknowledged our intuition as the content of the different pages is updated over time and, obviously, the more time spent, the more content might change.

Amazon and Google Images services have a smooth moderate decrease over time. The definition of an acceptable

[11]https://fortiguard.com/webfilter

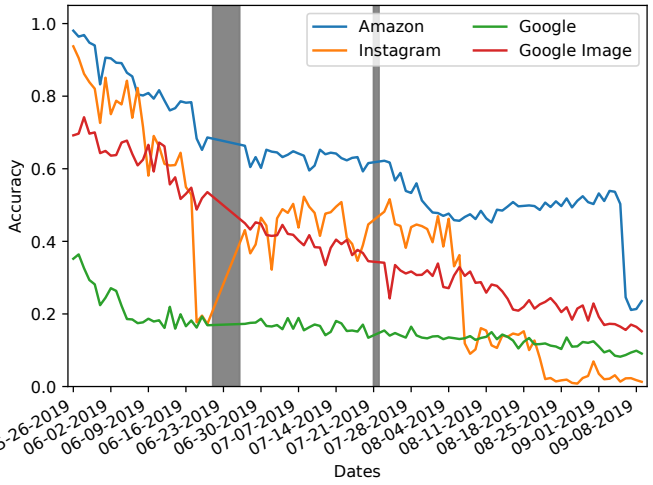

Fig. 2: Mean accuracy over time for each day

accuracy depends on the type of application. For example, if the objective is to block some users accessing illicit contents, high accuracy is desired. In case the objective is to extract a general trend of user activities, lower accuracy can be acceptable. Regarding Google search engine, we observe a fast decrease in the first days till a stable value around 0.2. Therefore, in addition to a higher difficulty of monitoring this service as highlighted in [1], the content seems to be more dynamic and invalidates the test of time for this particular case.

Instagram follows the same decreasing trend profile as Amazon and Google Images except three noticeable drops. We assume here two hypotheses: (1) the content of pages related to a monitored keyword has been modified or (2) the manner to send content has changed, for example using new library/software including updates at server-side. The first hypothesis was manually invalidated based on stored screenshots. No clear content update can be observed. Therefore, Instagram service might have been updated during our experimentation. Because accuracy was restored a few weeks after the first drop (mid June 2019), a plausible explanation is a restoration to a past configuration of the service. Amazon exhibits a similar drop toward the end of the period with the same plausible explanation of software changes altering the features.

B. Dynamic models

The previous experiment shows a decrease over a relatively low number of days. Although re-training the RF models can be done offline, we evaluate in this section an appropriate update frequency.

In this experiment, the models are updated every n_d days assuming the last 15 days of traces for learning (sliding window). In Figure 3, n_d is set to 5 and 10 days for comparison purpose. The vertical doted lines indicate the days when the models are updated. For readability, exact dates are not shown but are the same as in Figure 2.

In both cases, $n_d = 5$ or $n_d = 10$, retraining the models annihilates the impact of time on the classifier. As expecting, with a lower number of days n_d between training, the accuracy

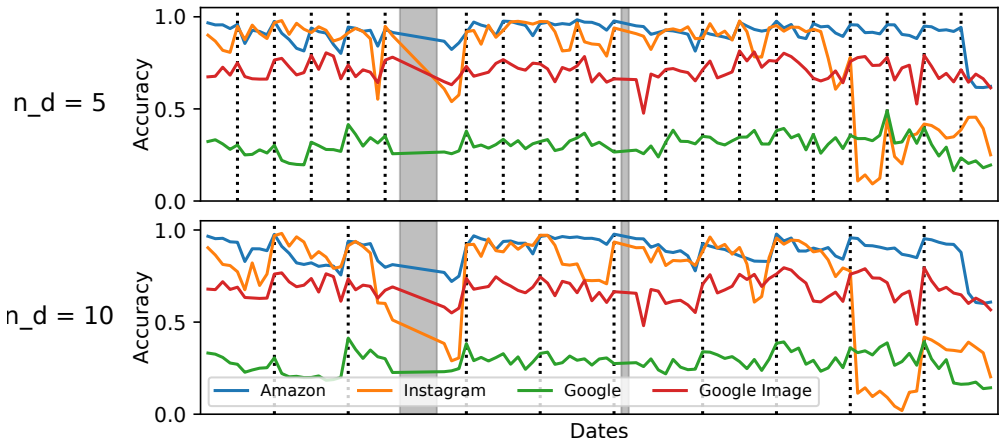

Fig. 3: Mean accuracy every day with new learning every n_d days

remains higher. The classification for Amazon, Google and Google Image are now highly stable over the time. From a practical point of view, a five day period is enough to collect new traces and retrain the model, especially knowing that learning the different models (one per each service) does not need to be synchronized on same days. The decrease observed over the time in Figure 2 is now compensated. For Instagram, the heavy drop in accuracy highlighted in Figure 2 are also easily corrected except for August 21.

In mostly all cases, re-learning the models regularly (about every week) with new captures can maintain the accuracy over the time.

C. Discussion

In general, accuracy in our current study is lower than our previous research work [1]. It is thus natural to identify and discuss the reasons that can explain this situation. We noticed two major differences between the two evaluations:

- The division of traces for training and testing in [1] are random but, for a given keyword, they are all captured successively in a few minutes. In this paper, a more realistic scenario is considered. The traces for learning and testing are collected over completely disjointed period of time. Actually, even a single day difference has an impact.
- The selected keywords in this paper have been voluntary defined based on trends and topicality in order to have more volatile content and evaluate our technique in a worst case scenario. It leads to different accuracy per single keyword as shown in Figure 4 focusing on the 10% worst and best keywords (keywords with the worst/best mean accuracy for the whole experiment). Although a large span in the accuracy can be observed, it also clearly shows that for a part of keywords (top 10%) our technique is very efficient even in the case of Google.

We expect that these observations are also helpful for future research in traffic analysis in order to better customize the scenarios, according to predefined criteria (worst case analysis, realistic case) and for researchers interested in our datasets.

V. Test-of-space

This section presents the evaluation of the classification for more than 3000 other services (test-of-space) based on the dataset presented in Section III-D.

As explained, a RF model is built for each service knowing that the built dataset provides 20 keywords per services. Because there are 20 traces by keyword, the learning stage is based on 16 traces while the remaining four are kept for testing.

The cumulative mean accuracy is reported in Figure 5 in comparison to a random classifier with an accuracy of 0.05 (i.e. select a random class out of 20). This figure shows that 50% of the websites have an accuracy higher or equals to 0.9. On the remaining 50%, 40% have an accuracy higher than 0.5.

Less than 2% of tested websites leads to an accuracy bellow the random cut-off. Through a manual analysis of the captured screenshots, two potential causes of misclassification have been identified:

- Whatever the specified keyword, the website always returns the same webpage. It is due to a non relevant keyword used or to an internal problem at the server side (*e.g.* maintenance in progress).
- The returned page is different for each keyword but the page is constituted of a very few objects although *H2Classifier* has been designed assuming complex webpages composed of numerous objects as presented in section II-B. For example, the news website *www.metro.us* search results are displayed as a simple text list without images. For the very same reason, accuracy highly differs between Google (search engine) and Google Images as shown in section IV.

Reminding there was no particular selection of tested websites and that for 50% of them, the accuracy is higher than 0.9, the evaluated classification technique is quite generic and can be used off-the-shelf for a large amount of services. It further confirms its efficiency which was solely evaluated on five services in [1].

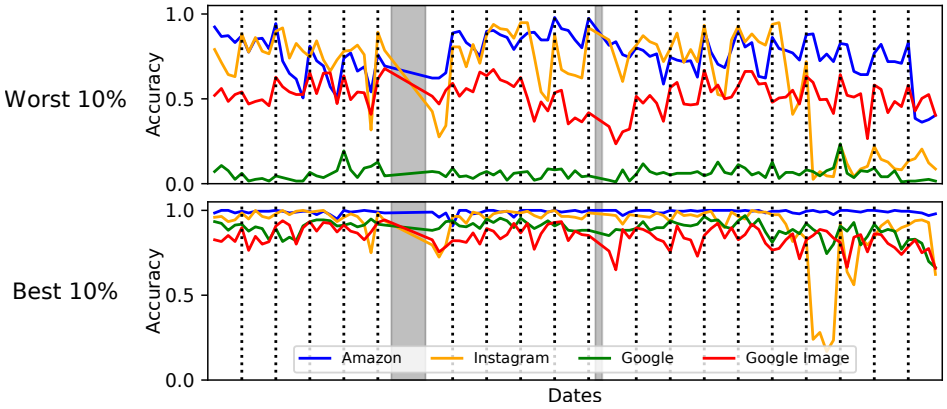

Fig. 4: Mean accuracy every day with new learning every 5 days for the 10% worst (top) and best (lower) keywords

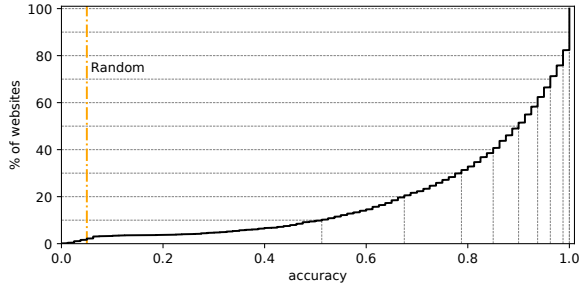

Fig. 5: Cumulative curve of accuracy for the 3096 websites

VI. Conclusion

In this paper, we evaluate the practical application of *H2Classifier* assuming its deployment in real scenarios where HTTPS traffic must be monitored over long time period and on many web services. Although *H2Classifier* has been specifically designed to monitor user activities in h2-based services (HTTP2 over TLS), the designed methodology and outcomes (results and discussions) are helpful for other encrypted traffic classification or fingerprinting techniques leveraging machine-learning. To summarize, the application of this type of technique is relevant and efficient for many different websites, assuming obviously the method has not been fitted to a particular one by design. However, the volatility of content in Internet as well as likely server software updates require regular re-learning, around every week in our case. As a future work, our plan is to evaluate if classification errors can be exploited to automatically detect falsified websites.

Acknowledgment

This work has been partially supported by the NATO Science for Peace and Security Programme under grant G5319 ThreatPredict, the European Union's Horizon 2020 research and innovation programme under grant agreement No 830927 and from the French National Research Agency (ANR), project number ANR-19-CE39-0011-01.

References

[1] P. Brissaud, J. Francçis, I. Chrisment, T. Cholez, and O. Bettan, "Transparent and service-agnostic monitoring of encrypted web traffic," *IEEE Trans. Network and Service Management*, vol. 16, no. 3, pp. 842–856, 2019.

[2] D. Wagner, B. Schneier *et al.*, "Analysis of the ssl 3.0 protocol," in *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996.

[3] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, 2015.

[4] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify https services," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2016.

[5] A. Pescape, A. Montieri, G. Aceto, and D. Ciuonzo, "Anonymity services tor, i2p, jondonym: Classifying in the dark (web)," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[6] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011.

[7] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*.

[8] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting." in *USENIX Security Symposium*, 2014.

[9] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique." in *USENIX Security Symposium*, 2016.

[10] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, Jan 2018.

[11] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, Jan 2016.

[12] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, "Effective and real-time in-app activity analysis in encrypted internet traffic streams," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017.

[13] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *10th USENIX Workshop on Offensive Technologies*. USENIX Association, 2016.

[14] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. ACM, 2017.

[15] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *Security and Privacy*. IEEE, 2008.

[16] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob?" in *USENIX Security Symposium*, 2007.

[17] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014.

[18] L. Breiman, "Random forests," *Machine learning*, 2001.

[19] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR*. ACM, 2015.